REMARKS

Applicant has amended independent claim 1 to include the feature, previously recited in dependent claim 2, that a bit mask is used for selectively loading the register, and to further clarify that each bit of the associated bit mask identifies a different byte of the register into which the data is loaded. Support for the clarification is found, for example, at page 11, lines 8-12. Independent claim 1 was also amended for greater clarity. Independent claims 10 and 19 were similarly amended. Claims 2, as well as claims 11 and 20, were cancelled. After these amendments, claims 1, 3-10, 12-19, and 21-27 are pending in the above-identified patent application. Claims 1, 10 and 19 are independent claims.

The examiner maintained the rejection of claims 1-8, 10-17 and 19-26 under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 4,569,016 to Hao in view of the reference "Computer Organization and Design" by Hennessey and Patterson, and further in view of the reference "Computer Architecture, a Quantitative Approach", also by Hennessey and Patterson. The examiner also maintained the rejection of claims 9, 18, and 27 under 35 U.S.C. §103(a) as being unpatentable over Hao in view of "Computer Organization and Design" to Hennessy et al., and further in view of U.S. Patent No. 5,832,258 to Kiuchi.

The examiner in response to applicant's September 7, 2005, Amendment in Reply to Office Action, stated at page 12, paragraph 37 of the Office Action:

> **Contrary to what Applicant argues, Hao teaches a mask that can specify multiple different combinations for selectively loading data into a specified register. The American Heritage Dictionary, 4th Ed., defines any as, 'One, some, every, or all without specification.' ... Since Hao teaches multiple combinations can be specified, Hao's mask qualifies as being capable of specifying "any combination" (Specifically, from the definition of any, multiple combinations falls under "one or some"). Furthermore, Applicants argument states, "Hao's mask cannot specify any combination of bits," which is clearly not true, since Applicant admits that Hao can specify combinations of bits in the paragraph directly above the argument on page 10 of the Applicant's "Remarks".**
>
> **c. Applicant submits an example following the argument, wherein Applicant's invention can selectively load bytes 1 and 3, whereas Hao cannot. However, this limitation is not required by the present claim language.**

Applicant's amended independent claim 1 recites the feature that the specified combination of the bytes is loaded to a transfer register "with loading using an associated bit mask, with each bit of the associated bit mask identifying a different byte of the transfer

register." For example, to selectively load data into a four (4) bye register, a 4-bit is used, where each bit identifies one of the bytes of the register. Thus, if the 1st and 3rd bytes of the register are to be loaded with data, the bit mask would be set to 0101.

In contrast, as explained in applicant's September 7, 2005, Amendment in Reply to Office Action, Hao discloses instructions that perform operations, such as a rotation operation, under control of a mask (Abstract). Particularly, with reference to its rotate instruction, Hao explains that "the result of the rotate instruction is either inserted into the register under control of the mask provided, or is AND'ed with the mask before being loaded into the register" "(Hao's col. 13, lines 1-4). The mask Hao uses is an 11-bit field in the instruction that indicates how to construct the mask that is to be used to AND the rotated operand into the register. The resultant constructed mask includes a central string of consecutive 1's or 0's, surrounded by 0's or 1's respectively. Bit 21 of the mask field in the instruction indicates whether the central string is that of 1's or 0's. The next 5 bits in the mask field indicate the left most position of the central string in a 32-bit register, while the last 5 bits of the mask field indicate the right most position of the central mask string in the a 32-bit register (col. 12, lines 7-23).

Thus, unlike applicant's mask, Hao's mask field (i.e., the field included with the instruction) does not identify particular bytes within a register that are to be loaded with data, but rather specifies how to construct a resultant mask that includes sequential strings of 1's and 0's. The constructed mask, in turn, also does not identify particular bytes within a register, but rather simply includes string sequences of 1's and 0's that are subsequently AND'ed with the data source. Accordingly, Hao does not disclose or suggest "with loading using an associated bit mask with each bit of the associated bit mask identifying a different byte of the transfer register," as required by applicant's independent claim 1.

Hennessey's "Computer organization and Design" generally describes the benefits of connecting multiple processors together (see page 712). However, Hennessy does not provide any technical details of any sort regarding the configuration and operation of such interconnected multiple processors.. Accordingly, Hennessey's "Computer organization and Design" does not disclose or suggest instructions or operations that cause the ALU to selectively load any specified combination of bytes of data within a transfer register. Therefore, Hennessey's "Computer organization and Design" also does not disclose or suggest "with loading using an

associated bit mask with each bit of the associated bit mask identifying a different byte of the transfer register," as required by applicant's independent claim 1.

Hennessy's "Computer Architecture A Quantitative Approach" describes a protection mechanism to protect processes' states during context switches by restricting different processes to operate within memory bounds (see page 448). Nowhere does Hennessy's "Computer Architecture A Quantitative Approach" describe instructions or operations that cause the ALU to selectively load any specified combination of bytes of data within a register. Hennessy's "Computer Architecture A Quantitative Approach", therefore, does not disclose or suggest "with loading using an associated bit mask with each bit of the associated bit mask identifying a different byte of the transfer register," as required by applicant's independent claim 1.

Since none of the cited references discloses or suggests, alone or in combination, the feature of "with loading using an associated bit mask with each bit of the associated bit mask identifying a different byte of the transfer register," claim 1 is therefore patentable over the cited references. Claims 3-9 depend from claim 1 and are therefore patentable for at least the same reasons as independent claim 1.

Independent claims 10 and 19 recite the feature of "selectively loading any combination of bytes ... ,using an associated bit mask with each bit of the associated bit mask identifying a different byte of the register," or similar language. For reasons similar to those provided with respect to independent claim 1, at least this feature is not disclosed by the cited art. Accordingly, independent claims 10 and 19 are patentable over the cited art. Claims 12-18 depend from independent claim 10 and are therefore patentable for at least the same reasons as claim 10. Claims 21-27 depend from independent claim 19 and are therefore patentable for at least the same reasons as independent claim 19.

Additionally, as noted above, the examiner rejected claims 9, 18, and 27 under 35 U.S.C. §103(a) as being unpatentable over Hao in view of "Computer Organization and Design" to Hennessy et al., and further in view of Kiuchi.

The examiner admitted that Hao and Hennessy's "Computer Organization and Design" do not teach "an optional token that is set by a programmer and specifies to set arithmetic logic unit (ALU) condition codes based on the result" (page 10, paragraph 33 of the November 2,

2005, Office Action). The examiner, however, argued that Kiuchi describes this feature. Applicant respectfully disagrees.

Kiuchi describes instructions that include condition code fields that identify predefined conditions and also identify whether a condition code register should be updated when the data processing operation is performed by the execution unit of a digital signal processor (Abstract). For example, the condition code 0001 indicates that the condition code register (ccr) is not to be updated with the flags generated during the execution stage of the current instruction (col. 37, line 59-61). On the other hand, the condition code 0000 indicates that the condition code register is to be updated during the execution stage of the current instruction (col. 37, lines 56-58).

Table 1 at Kiuchi's columns 37-38 further describes that the condition codes 0010-1111 identify the condition flag status (namely, one of the Carry (C), Negative (N), Overflow (V), Zero (Z), Carry or Borrow (GC), or Over Range (VR) flags) based upon which a determination is made of whether the current instruction should execute. For example, if the condition code 0010 is used with a particular instruction, that instruction will execute only if the value of the Carry (C) flag is 0 (col. 37, lines 62-64). When the condition codes 0010-1111 are used, the condition code register is not updated.

But nowhere in Kiuchi's Table 1, or elsewhere in Kiuchi, does Kiuchi disclose or suggest an instruction that "comprises an optional token that is set by a programmer and specifies to set arithmetic logic unit (ALU) condition codes based on the result formed in the ALU," as required by applicant's claim 9. Further, the condition codes or flags that are set by the ALU after execution of instructions that specify examination of condition codes fails to suggest the claimed option token, since the optional token is specified by the programmer.

Since none of the reference cited by the examiner discloses or suggests, alone or in combination, the feature of "the instruction further comprises an optional token that is set by a programmer and specifies to set arithmetic logic unit (ALU) condition codes based on the result formed in the ALU," applicant's claim 9 is thus patentable over the cited art.

Claims 18 and 27 recite "an optional token that is set by a programmer and specifies to load arithmetic logic unit (ALU) condition codes based on a result formed in the ALU", or similar language. For similar reasons as those provided with respect to claim 9, at least this

feature is not disclosed or suggested by the cited art. Accordingly, applicant's claims 18 and 27 are also patentable over the cited art.

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

Canceled claims, if any, have been canceled without prejudice or disclaimer.
Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

No fees are believed due. Please apply any other charges to deposit account 06-1050, referencing attorney docket 10559-309US1.

Respectfully submitted,

Date: Jan. 5, 2006

Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906
21240561.doc